TASK: PA12
CDRL: M002R1
24April 1995



# Domain Architecture Handbook and Testbed Lessons Learned




STARS-AC-M002R1/001/00

**TASK: PA12**
**CDRL: M002R1**
**24 April 1995**

# INFORMAL TECHNICAL REPORT

For

# SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS)

*Domain Architecture Handbook and Testbed*
*Lessons Learned*

STARS-AC-M002R1/001/00
24 April 1995

CONTRACT NO. F19628-93-C-0130



Prepared for:
Electronic Systems Center
Air Force Systems Command, USAF
Hanscom, AFB, MA 01731-2816

Prepared by:
Unisys Corporation
12010 Sunrise Valley Drive
Reston, VA 22091

**TASK: PA12**
**CDRL: M002R1**
**24 April 1995**

Data Reference: STARS-AC-M002R1/001/00
INFORMAL TECHNICAL REPORT
*Domain Architecture Handbook and Testbed*
*Lessons Learned*

Distribution Statement "A"
per DoD Directive 5230.24
Authorized for public release; Distribution is unlimited



This document, developed under the Software Technology for Adaptable, Reliable Systems (STARS) program, is approved for release under Distribution "A" of the Scientific and Technical Information Program Classification Scheme (DoD Directive 5230.24) unless otherwise indicated. Sponsored by the U.S. Advanced Research Projects Agency (ARPA) under contract F19628-93-C-0130, the STARS program is supported by the military services, SEI, and MITRE, with he U.S. Air Force as the executive contracting agent. The information identified herein is subject to change. For further information, contact the authors at the following mailer address:
delivery@stars.reston.unisysgsg.com.

**TASK: PA12**
**CDRL: M002R1**
**24 April 1995**

Data Reference: STARS-AC-M002R1/001/00
INFORMAL TECHNICAL REPORT
*Domain Architecture Handbook and Testbed*
*Lessons Learned*

## Abstract

This document describes the lessons learned from STARS task PA12. Under this task, the Unisys STARS team developed a testbed for analyzing *architecture description languages* (ADLs) and a World Wide Web "Software Architecture Technology Guide" to provide the DoD software engineering community with a broad range of information about software architecture technology. The lessons learned focus on issues associated with:

- Developing World Wide Web documents in general and the Software Architecture Technology Guide in particular.
- Establishing criteria for analyzing ADLs and developing scenarios for applying the criteria.
- Obtaining ADL technologies for analysis and analyzing them using the criteria and scenarios.

**TASK: PA12**
**CDRL: M002R1**
**24 April 1995**

Data Reference: STARS-AC-M002R1/001/00
INFORMAL TECHNICAL REPORT
*Domain Architecture Handbook and Testbed*
*Lessons Learned*

**Principal Author(s):**

Frank Svoboda — 24 Apr 95

*Frank Svoboda* — Date

Richard E. Creps — 27 Apr 95

*Dick Creps* — Date

*Paul Kogut* — Date

**Approvals:**

Teri F. Payton — 4/27/95

Program Manager *Teri F. Payton* — Date

(Signatures on File)

# REPORT DOCUMENTION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY *(Leave Blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | 24 April 1995 | Informal Technical |

**4. TITLE AND SUBTITLE**
Domain Architecture Handbook and Testbed Lessons Learned

**5. FUNDING NUMBERS**
F19628-93-C-0130

**6. AUTHOR(S)**
Frank Svoboda, Dick Creps, Paul Kogut

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Unisys Corporation
12010 Sunrise Valley Drive
Reston, VA 22091-3499

**8. PERFORMING ORGANIZATION REPORT NUMBER**
CDRL NBR
STARS-AC-M002R1/001/00

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Department of the Air Force
ESC/ENS
Hanscom AFB, MA 01731-2816

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
M002

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Distribution "A"

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

This document describes the lessons learned from STARS task PA12. Under this task, the Unisys STARS team developed a testbed for analyzing *architecture description languages* (ADLs) and a World Wide Web "Software Architecture Technology Guide" to provide the DoD software engineering community with a broad range of information about software architecture technology. The lessons learned focus on issues associated with:

- Developing World Wide Web documents in general and the Software Architecture Technology Guide in particular.
- Establishing criteria for analyzing ADLs and developing scenarios for applying the criteria.
- Obtaining ADL technologies for analysis and analyzing them using the criteria and scenarios.

**14. SUBJECT TERMS**

**15. NUMBER OF PAGES**
16

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | SAR |

Data Reference: STARS-AC-M002R1/001/00
INFORMAL TECHNICAL REPORT
*Domain Architecture Handbook and Testbed*
*Lessons Learned*

# Table of Contents

# Foreword

The organization of this document reflects the structure of task PA12. The Introduction (Section 1) gives the motivation for the overall task. The Task Overview (Section 2) introduces the major objectives of the task and the three subtasks that were performed to achieve these objectives. Each of the subtasks is addressed in more detail in Sections 3, 4, and 5, with a description of the individual approaches, implementation issues, tool use, and conclusions, as appropriate. The overall conclusions for the task are given in Section 6. The document concludes with a list of relevant acronyms and a list of bibliographic references.

The audience for the products of the work described herein is defined as the DoD Software development community. Although the testbed results are of particular interest to persons with an acknowledged interest in software/system architectures and their representation, the introductory sections of the World-Wide Web-based Software Architecture Technology Guide may be used even by those with no current, perceived need. It is our intention to provide resources that will assist organizations in making informed decisions about software architecture representation technology.

# 1. Introduction

Architecture-based reuse is one of the major themes of STARS [STARS93] and of the DoD Software Reuse Initiative Vision and Strategy[DoD92]. Domain architectures are presented as the key technical foundation for systematic reuse. Yet architecture representation to support reuse remains among the most difficult technical issue to overcome. Several ARPA and Software Reuse Initiative programs have begun to address architecture representation issues, but more work is needed, and some means is needed for sharing, assessing, and comparing the results of these programs and other relevant DoD efforts.

The STARS demonstration projects are each using distinct techniques for representing and applying domain architectures, reflecting interpretations of the STARS concept of megaprogramming. For example, the domain engineering effort underway on the Army STARS demo project in the ELPA domain is producing an initial means for representing a domain architecture through application of the ODM and GenVoca methods for domain modeling and architecture definition, respectively.

CARDS has applied an architecture representation approach as a basis for their Command Center Library and their prototype system composition capability. CARDS also conducted an architecture workshop in November 1993 and has developed an architecture tutorial which has been presented in several forums. Other DoD efforts, such as the SEI model-based software engineering project and the ARPA DSSA, Prototech, and Software Composition programs, are also exploring architecture representation issues and are developing specific techniques (e.g., OCA, LILEANNA, Micro-Rapide, UniCon). In addition, tools and techniques have been produced by other government agencies (e.g., the NASA-sponsored KAPTUR tool -- now being commercialized by CTA as "Capture") and have begun to appear in the commercial marketplace (e.g., TRW's UNAS/SALE products, derived from work originally funded by the Air Force).

These assorted capabilities vary widely in terms of their level of current practicability, their applicability to different classes of problems, and so on. For example, UNAS/SALE, KAPTUR, and some of the SEI and CARDS techniques have already been applied in practice, while the ARPA DSSA and Prototech programs are for the most part taking more formal, language-based approaches that will bear fruit in the longer term.

The DoD SW development community needs access to resources that can provide assistance in understanding architecture issues and choosing appropriate architecture representations. A handbook-style document is needed (preferably in an on-line, easy-to-digest format, such as World-Wide Web hypermedia) that gives the DoD community an overview of the conceptual underpinnings of domain-specific architectures and provides a reference for understanding the techniques that have been or are being developed and how they can be applied. In conjunction with the development of such a document, a testbed is needed to objectively assess the extent to which these techniques are ready to be applied in practice and the kinds of problems to which they are most applicable. Such a testbed can also assist in identifying broad deficiencies in available techniques and establishing future research priorities.

# 2. Task Overview

This task had as its three primary objectives the development of:

1. A set of *scenarios* for analyzing ADL properties, based on criteria established in the ADL descriptive model framework (a taxonomy of ADL features) being developed by the SEI and CARDS. The scenarios, and the results from applying them, are intended to help the DoD software development community understand, assess, and select appropriate architecture representation techniques.

2. A *testbed environment*, populated with ADL support tools, in which the scenarios can be applied to analyze specific ADLs.

3. A web-based *Software Architecture Technology Guide* — an on-line resource for understanding architecture technology issues; among other things, it is designed to include the results of specific ADL analyses conducted under this task.

For a more detailed treatment of the scenarios developed for the testbed and the results of applying them to specific ADLs, please consult the other report produced under this task, *Scenarios for Analyzing Architecture Description Languages* [STARS95]. To explore the Software Architecture Technology Guide, see the following URL:

http://www.stars.reston.unisysgsg.com/arch/guide.html

## 2.1. ADL Analysis Scenarios

An initial set of scenarios for analyzing ADL properties has been developed. Each scenario is designed to reveal the presence or absence of specific ADL features defined in the SEI/CARDS feature model. The scenarios involve activities addressing both architecture creation (e.g., by a domain engineer defining a domain architecture) and architecture utilization (e.g., by an application engineer instantiating a domain architecture for use in a specific system). The scenarios emphasize empirical analysis through operational usage of the notations and tools, rather than through static application of a list of criteria.

## 2.2. ADL Testbed Environment

To enable analysis of a specific set of ADLs and supporting tools, an initial ADL testbed environment has been hosted on a Sun workstation at the STARS Technology Center. This testbed has been populated with a set of ADL support tools that were the candidates for hands-on analysis. Of these candidates, two ADLs/tools were analyzed using the scenarios in conjunction with the SEI/CARDS feature model and the results were documented.

## 2.3. Software Architecture Technology Guide

A World-Wide Web (WWW)-based Software Architecture Technology Guide (hereafter often

referred to as the Guide or the "web node") has been developed that provides an introduction to software architecture concepts and terminology and includes summaries and analyses of a representative sampling of ADLs addressing both the state-of-the-practice and the state-of-the-art. It includes the ADL analysis scenarios and the results from applying them to specific ADLs.

The Guide is packaged in a "self-guided tour" format, enabling the user to easily navigate among the major topics it addresses. The Guide itself offers a significant amount of information about these topics, but it also provides links to numerous external on-line resources such as DoD architecture-focused program home pages and architecture-related reference materials. As a result, the Guide offers the DoD software engineering community a unique mixture of a broad range of architecture-related information, packaged to support easy on-line browsing and downloading.

# 3. Implementation — ADL Analysis Scenarios

## 3.1. Approach

Our overall scenario development approach was inspired by the software engineering environment evaluation methodology developed by Weiderman, et al, at the SEI [Weid86] and subsequently applied and refined by Feiler and Smeaton [Feil88], Christie [Chri94], and others. Rombach's process representation assessment framework [Romb91] also influenced our approach. The Weiderman methodology is founded on several key principles, including the following:

- Evaluations should be based on the results of well-defined experiments to maximize their objectivity and repeatability.
- Evaluation methods should clearly define the scope of the functionality to be evaluated and should focus on core functionality that will be broadly relevant to the evaluated technologies and of greatest interest to the evaluators.
- Methods should be based on general user activities rather than detailed tool operations.
- Methods should be as independent of the technologies to be evaluated as possible to minimize potential biases.
- Methods should be extensible to accommodate additional user activities and experiments.

Our testbed development approach involved four major activities reflecting these guiding principles. These activities were:

1. Establish criteria to use as a basis for assessing key ADL properties.

   In this activity we leveraged the ADL feature analysis work being performed jointly by the SEI and CARDS [Clem95, KC95]. In this work, a simplified ODM domain analysis

was performed on the ADL domain. The analysis has yielded a descriptive model framework (or "feature model") which defines a taxonomy of properties ("features") that ADLs can possess. Some of these features are amenable to static analyses that needn't involve scenario-based hands-on experiments to yield reasonably objective results. We thus prioritized the features in terms of (a) how difficult they are to analyze statically and objectively and (b) how practically relevant they are to ADLs currently available for analysis. Examples of the ADL features we elected to emphasize are:

- Support for various architecture styles
- Ability to represent architectures of various categories of systems (e.g., real-time, distributed)
- Understandability of architectures described using the ADL
- Modifiability of architectures described using the ADL
- Support for variability within an architecture description (e.g., within component interfaces) to facilitate reuse
- Scalability to support large-scale architectures or components

2. Develop scenario-based methods for objectively applying the criteria to ADLs to determine their properties.

   Our approach emphasizes analysis through scenario-based operational usage of the ADLs and supporting tools relative to the selected criteria, supplemented by static analysis of the ADLs relative to other features in the SEI/CARDS feature model that are more amenable to such analyses. Included with the feature model is a detailed form for recording the results of an ADL analysis, feature by feature. We adopted this form as the medium for recording the results of our analyses.

   Our scenarios are founded on a set of "model problems" for architecture definition (inspired by Shaw's work in this area [Shaw94a]) that concisely pose interesting and relevant challenges to ADLs. The problems have been defined to exercise the range of ADL features from which the assessment criteria are derived. Specifically, we have defined two scenarios based on model problems in the automobile cruise control and military command center domains. Selection of the cruise control domain was inspired by Shaw's comparative analysis of cruise control architectures [Shaw94b], while selection of the command center domain was influenced primarily by the PRISM and CARDS work that has been done in this area.

## 3.2. Work-Products

The work-products for this effort are the ADL analysis scenarios and the corresponding evaluation information. The scenarios focus on two model problems in two domains/architectural contexts. Each scenario potentially provides a counterpoint against biases introduced by the

other.

1. Cruise Control Scenario — problem area is well-known, with many existing solution architectures, in several different styles; This scenario may have more appeal to the researcher, given its manageable level of complexity.

2. Command and Control (C2) Scenario — this scenario addresses a problem area that may be of more interest to practitioners. Actual experience on the Air Force CARDS and PRISM programs in this application area was utilized in the development of the C2 scenario.

As noted above, the details of the scenarios can be obtained in the other PA12 technical report, *Scenarios for Analyzing Architecture Description Languages.*

## 3.3. Issues and Lessons Learned

- **Architecture style proved an effective feature-of-focus.**

  Architecture style was chosen as the key feature-of-focus for making decisions about the nature of the model problems. It was determined to be the feature most likely to introduce the most representational bias into a scenario, so it was important to emphasize that feature in order to better manage and predict the bias. One way to mitigate the bias was to design the scenarios to address a wide range of styles. In executing the scenarios, we found that the topological notion of "shape" of architectural connection models was an excellent starting point for understanding the basic concepts of architecture styles.

- **Evaluation criteria are subject to interpretation and subjectivity**

  In certain cases, the assessment of an ADL with respect to given attributes was subject to interpretation and potential misunderstanding. In one case, the terms "Cross-reference" and "translation" suffered from perceived ambiguity among team members. Multiple-choice responses were particularly prone to variation. Although consensus was easily achieved by our small team, it may be more elusive for a larger sample of the population (see next two items).

- **Scenario language should reflect the variability in the ADL domain and not introduce undue biases.**

  As an example, the use of the word "statement" has little meaning in the context of the evaluation of a graphical ADL. The term "construct" was adopted in its place.

- **Assumptions should be explicitly identified.**

  Analysis questions based on assumed semantic evaluation capabilities are not applicable to ADLs without those capabilities. For example, tolerance of incomplete models exists both in those ADLs with built-in tolerance and those incapable of determining completeness.

- **Hands-on experience in evaluating architecture representations using the scenarios has**

**has yielded valuable feedback for improving them.**

We learned a lot about the inadequacies of the informal architecture description in the scenarios by trying to model it with the ADL (especially UniCon). The descriptions should be refined based on this experience. Doing the evaluation helped immensely in clarifying the ADL descriptive model features but there are still many areas especially in the area of tools which need additional development and clarification. This can only be done the hard way — by hands-on evaluation of more ADLs.

## 3.4. Conclusions

We believe that our selection of model problems and ADLs have led to scenarios that yield comparable results across ADLs, problems, and architectural styles. Scenarios can be completed with a minimum of creative activity and the time and effort for completion is relatively predictable. Working through the scenarios have the additional benefit of forcing the evaluator into a disciplined analysis of his own architecture description requirements.

# 4. Implementation — ADL Testbed Environment

## 4.1. Approach

Our approach to developing and exercising the testbed environment involved the following two major activities:

1. Select a representative set of ADLs and supporting tools to analyze, and obtain and install the tools to support the hands-on, scenario-based analyses.

   We analyzed two ADLs during the course of our task. This number was limited primarily by resource constraints on the task. We planned to select ADLs that covered a range of practical and research interests. To a certain extent, we have achieved this goal with the analysis of CMU's UniCon (a research prototype) and CTA's soon-to-be commercial offering Capture.

   The installed tool base, in conjunction with the scenario-based analysis methods and underlying ADL feature model, together constitutes the ADL analysis testbed.

2. Apply the methods and criteria to the selected ADLs/tools and document the results.

   We analyzed the selected ADLs both statically (for those criteria that are amenable to such analysis) and more dynamically through application of the scenarios. The scenarios are designed to yield results that are objective, repeatable, and comparable, and we enacted them to preserve these design principles as much as possible. During scenario enactment, we recorded results about the ADL features in the form supplied with the SEI/CARDS feature model. We also recorded notes and rationale about the ADLs and workproducts during scenario enactment and noted issues and lessons learned that may help improve the

methods in the future.

The SEI and CARDS have already performed static analyses of a small set of ADLs to help validate the feature model. For further validation, they have also asked a number of ADL designers and users to characterize their ADLs in terms of the feature model. We collected and interpreted some of these findings in performing our own static analyses.

## 4.2. Work-Products

The work-products for this effort are the installed tool base portion of the testbed and the results of the ADL analyses. The tool base includes the executables and architecture models for both Capture and UniCon. The UniCon kit also includes the Odin enhanced "make" utility, which was required for its installation.

The detailed results of the ADL analyses are published in the other PA12 technical report and in the Software Architecture Technology Guide.

## 4.3. Issues and Lessons Learned

- **Dependencies upon commercial products prevented the acquisition of certain ADL products.**

  Our choice of ADLs was dependent on the availability of supporting tools. Prerequisites for tool acquisition included no- or low-cost licensing and availability within the task completion time-frame. Several tools required commercial languages with accompanying GUI products such as Lisp or SmallTalk that priced them beyond our ability to obtain them. It should be noted that the absence of GUI components did not always preclude the use of textual tools such as parsers and editors.

- **The task's tight schedule limited tool acquisition.**

  The task's short duration made us especially susceptible to time-dependent requests for tools and supporting documents. The time deficit was compounded by holiday and end-of-year activities for both military and academic organizations, although we initiated solicitations early. We found that personal contact (either face-to-face or by phone) was most effective for support material acquisition. E-mail proved useful for confirmation of verbal communications and for providing a document trail for recording activities and progress. Not surprisingly , there also appears to be a strong correlation between stake in the use of tools/outcome of effort and supplier responsiveness.

The lessons learned from applying the scenarios are incorporated into the lessons under the scenario development task above, since those lessons directly relate to the quality and usability of the scenarios.

## 4.4. Conclusions

Our tool base was adequate for executing the evaluation scenarios on an adequately representa-

tive sample of ADLs and for providing feedback which was then used to perform interim scenario improvements. Although we would have preferred to have had more potential selections of ADL tools, the attendant learning curve for additional tools would have forced us eliminate all but several candidates. The results of our tool acquisition were a good fit to our time and manpower resources.

# 5. Implementation — Software Architecture Technology Guide

## 5.1. Approach

The Software Architecture Technology Guide was built under the following guidelines:

- The Guide will provide information on a representative set of ADLs that would be candidates for evaluation/ study in the testbed.
- New, original material will be minimized, except to provide an overview for each major topic and a structure for connecting the topics and accessing external material. The Guide will establish hypermedia links to existing material from ADL suppliers to provide easy access to information describing a cross-section of ADLs, ranging from the state-of-the-practice to the state-of-the-art.
- Significant technical papers will be included, either locally or via remote hypermedia links. These papers will be converted as necessary to either PostScript or HTML formats.

## 5.2. Tech support and tools

The following tools were integral in developing the Software Architecture Technology Guide:

- *Mosaic* — public domain web browser; *Mosaic* supports standard HTML and in-line graphic display in GIF format. Although our web work was performed using Mosaic, the *Netscape* browser is currently gaining in popularity and supports extensions to standard HTML, as well as the ability to display JPEG graphics formats.
- *FrameMaker* — Unix-based desktop publishing software by Frame Technology Corporation
- *Frame2html* — converts FrameMaker documents to HTML format. Frame2html is freeware available from Norwegian Telecom Research by anonymous ftp at

  bang.nta.no:pub/fm2html.tar.v.0.n.m.Z

- *xv* — Unix-based graphic utility; displays GIF, TIFF, pbm, Sun rasterfile, X11 Bitmap, PostScript, BMP, IRIS, JPEG, and PM bit-mapped (raster) formats. *xv* will also convert a graphic between any of these given formats. *xv* also performs color modification by changing RGB colors of any selected value to any other. Special effects include blurring (effective for size reduction), embossing, edge detection, oil painting simula-

tion, and image cropping.

- Screen Dump/Capture — various Unix utilities are available to capture graphic screen images and ultimately convert these to GIF format for inclusion within a Mosaic-readable web node.

- *gift* — transparent GIF converter (i.e., makes the background color of a GIF graphic transparent)

## 5.3. Work-Products

The work-products for this subtask are the text and graphics files which, when interpreted by a suitable browser, display the Software Architecture Technology Guide. The form and content of the Guide borrowed heavily from material in the CARDS Software Architecture tutorial (developed primarily by Kogut and Wallnau). STARS presentation materials were reused both for their textual and graphic content.

The technical papers hosted locally with the Guide (and the format conversions required to make them viewable) are also a result of this effort.

## 5.4. Issues and Lessons Learned

- **Configuration Management (CM) becomes more important as a web node grows in size and complexity.**

  As the count of the Guide's files grew (approximate count = 20), data management became more important. Backups should be made regularly to protect against catastrophic loss. Versioning could even be considered to map the evolution of a web node's content.

- **Standards for creation of web pages should be established, published, and reused.**

  Standards can establish a consistent "look and feel" for pages within a given web node and for multiple nodes within an organization. Such standards can also assist in the training of web presentation concepts.

- **Presentation esthetics play a significant role in acceptance of web media.**

  A significant difference between hypermedia web nodes and their sequential text-based counterparts (e.g., ftp, gopher sites) is the use of color and graphics. Even though the information content of the guide was considered of primary importance, much of the comment we received was on the presentation of the material. The full color and graphics capabilities of web presentations can be used to capture the interest of viewers as well as to focus their attention on desired areas.

  Our graphic banner (Figure 1) makes the analogy between software and classical architectures and appears at the top of subsequent pages. The banner identifies the Guide and helps clarify context in the navigation of multiple web nodes.


**Figure 1: Graphic Banner**

The image-mapped graphic index (Figure 2) depicts the virtual organization of the node and serves as its main navigation tool. The visual image of a blueprint helps reinforce the connection between more conventional notions of architecture and their analogs in software technology.

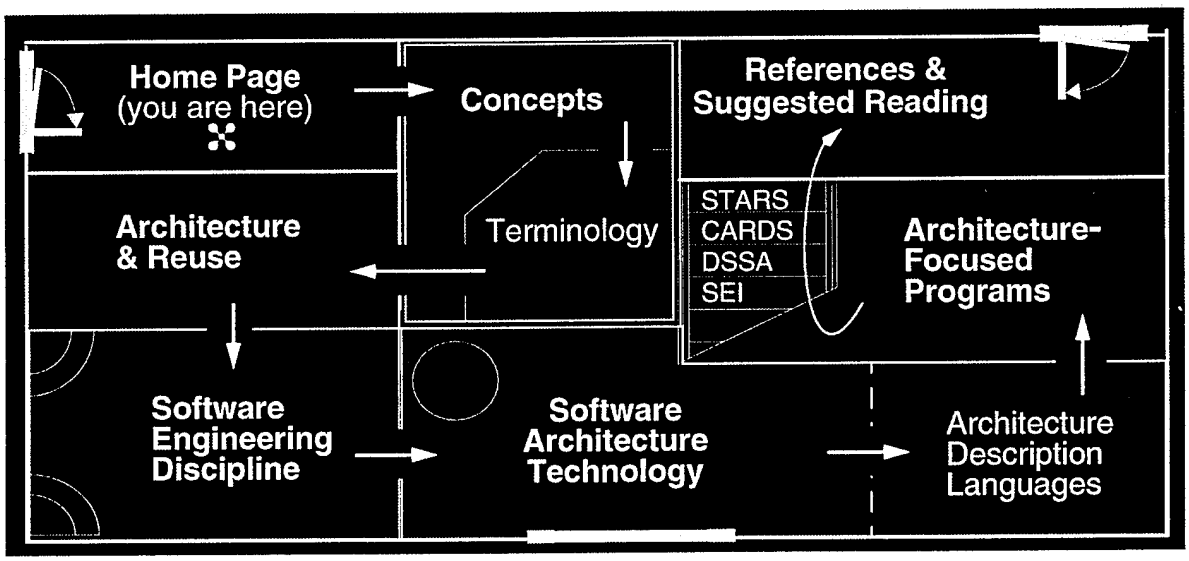


**Figure 2: Graphic Index**

The graphics tools available to us were of limited capability (primarily Framemaker). Even limited graphics tool can be enhanced with the use of clip art extracted from the web. We employed web search engines such as Web Crawler and the World-Wide Web Worm to search for additional graphics sources. Those anticipating extensive web work would be well-served by more advanced graphics tools such as CorelDraw or Micrografix Designer and a substantial clip-art library.

A consistent appearance (encompassing layout, choice of color, navigation, footnotes, etc.) is desirable. Decisions regarding appearance extended beyond this task to other STARS web

nodes. Although authors would like to plan how web pages are to be viewed, it is impossible to dictate how much users will see in a given viewing (i.e., the size of individual pages). Displays and personal window selections will vary in size.

- **Copyright issues exist for web usage of published materials.**

  There are varying constraints on the use of previously published materials in web nodes. Governing organizations, particularly the ACM and IEEE, have established (often conflicting) standards for the use of documents published in their name. These standards are still evolving. The URLs for access to the IEEE and ACM publishing policies are given below:

  **IEEE**: http://www.computer.org/publications.html

  **ACM**: http://www.acm.org/

- **Web resources for acquisition and distribution of materials should be identified on an on-going basis.**

  Government sources are not centralized. Restricted access to related nodes (e.g., ARPA Prototech, DSSA, and Software Foundations) hindered progress. Military web sites are particularly resistant to web search engines. Web creators also need to identify links and distribution sites for material upon release approval.

- **Web node feedback is almost non-existent unless directly solicited.**

  The comments ("gripes") utility provided on our home page has not been utilized by the public. This option allows readers to send e-mail comments to the authors of the web page. While we have received significant feedback on the node, responses were the result of personal contact with specific individuals.

- **Conversion utilities vary in their effectiveness.**

  The conversion utilities available from web sources can perform the following text conversions to HTML format adequately (though not well):

  - Word Perfect → RTF → HTML is adequate.
  - LaTex → HTML is much better, due to hierarchical structure inherent in LaTex.

## 5.5. Conclusions

Continued use of web-based development opens up possibilities for the automation of that development. Using text templates, decision models, and execution scripts for the purposes of automated web node creation would be relatively inexpensive and quite useful if more WWW work is probable. Given the burgeoning usage of the web, we might reconsider how we develop presentations with an eye towards their reuse in web applications. Slides are typically developed in landscape mode, whereas web documents are more suited to a portrait format. Graphics and accompanying text often needed to be sized manually to fit the new format. We may want to anticipate reuse of slides in the web and size accordingly. We also need to consider the differ-

ences between presentation of web material to live audiences and individual viewing, especially in the choice of print fonts and colors.

# 6. Overall Conclusions

The testbed will continue to yield valuable results to both the DoD software practitioner and research communities by:

- Providing practical methods for analyzing ADLs so that practitioners and researchers will have a concrete basis for conducting their own ADL analyses.
- Publishing the results of initial analyses of specific ADLs, which may be useful to practitioners who need to make near-term decisions about architecture support technology.
- Providing a basis for comparing and identifying deficiencies in available technology to help establish future research priorities.
- Establishing a baseline for future ADL analysis methods and criteria.

Additionally, this task is providing a useful new information resource to the DoD community in the form of the Software Architecture Technology Guide. The Guide collects a wide range of information about architecture technology and puts it at the fingertips of the Internet user to help broaden awareness of this increasingly important topic. The Guide has already generated significant interest among diverse software development communities (e.g., architecture, reuse, reengineering) in academia, industry, and government.

The challenges we have faced in developing the testbed are, in their full generality, immense. We believe that, despite the limited resources available to us in this task, we have established a useful set of initial capabilities. We recognize, however, that more work will be needed to generalize and extend our results.

# Acronyms

| | |
|---|---|
| **ADL** | Architecture Description Language |
| **ARPA** | Advanced Research Projects Agency |
| **BMP** | (Microsoft) BitMaP; a raster graphics imaging format |
| **CARDS** | Comprehensive Approach to Reusable Defense Software |
| **C2** | shorthand for "Command and Control" |
| **DoD** | Department of Defense |
| **DSSA** | Domain-Specific Software Architectures; ARPA-funded project for the development of architecture technology in fields of interest to the DoD |
| **ELPA** | Emitter Location and Processing Analysis; an electronic warfare (EW) application domain |
| **ftp** | (Unix) file transfer protocol |
| **GIF** | Graphic Interchange Format; a raster graphics imaging format — the only format currently supported by the Mosaic browser |
| **GUI** | Graphical User Interface |
| **HTML** | Hypertext Markup Language; the hypertext language that forms the basis for the World Wide Web |
| **HTTP** | HyperText Transport Protocol |
| **JPEG** | a compressed raster graphics imaging format |
| **IRIS** | a raster graphics imaging format |
| **KAPTUR** | Knowledge Acquisition for Preservation of Trade-offs and Underlying Rationale |
| **OASD** | Office of the Assistant Secretary of Defense |
| **ODM** | Organization Domain Modeling |
| **PM** | a raster graphics imaging format |
| **PRISM** | Portable, Reusable Integrated Software Modules |

| | |
|---|---|
| **RTF** | (Microsoft) Rich Text Format; a textual format specification |
| **SALE** | System Architect's Life-cycle Environment |
| **SEI** | Software Engineering Institute |
| **STARS** | Software Technology for Adaptable, Reliable Systems |
| **TIFF** | Tagged Image File Format; a raster graphics imaging format |
| **UNAS** | Universal Network Architecture System |
| **WWW** | World-Wide Web (also "the web") |

# References

**[Booch86]** Booch. "Object-Oriented Development," *IEEE Transactions on Software Engineering*, February 1986.

**[Chri94]** Christie. "A Practical Guide to the Technology and Adoption of Software Process Automation," CMU/SEI-94-TR-007, Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA, March 1994.

**[Clem95]** Clements, Kogut. "Features of Architecture Description Languages," *Proceedings of the Seventh Annual Software Technology Conference*, Salt Lake City UT, April 1995.

**[Crep95]** Richard Creps, Paul Kogut, Frank Svoboda. "A Testbed for Analyzing Architecture Description Languages," *Proceedings of the Seventh Annual Software Technology Conference*, Salt Lake City UT, April 1995.

**[DoD92]** DoD Software Reuse Vision and Strategy, Document #1222-04-210/40, July 1992.

**[Feil88]** Feiler, Smeaton. "The Project Management Experiment," CMU/SEI-88-TR-7, Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA, July 1988.

**[GS93]** Garlan, Shaw. "An Introduction to Software Architecture," CMU/SEI-93-TR-33, Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA, December 1993. Also in *Advances in Software Engineering and Knowledge Engineering, Volume I*, Ambriola and Tortora (eds.), World Scientific Publishing, Singapore, 1993.

**[KC95]** Kogut, Clements. "Features of Architecture Description Languages," draft SEI technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA, December 1994.

**[Romb91]** Rombach. "A Framework for Assessing Process Representations," *Proceedings of the 6th International Software Process Workshop: Support for the Software Process,* Hakodate, Hokkaido, Japan, July 1991.

**[Shaw94a]** Shaw, Allen, Garlan, Klein, Ockerbloom, Scott, Schumacher. "Candidate Model Problems in Software Architecture," unpublished manuscript, version 1.2, November 1994.

**[Shaw94b]** Shaw. "Making Choices: A Comparison of Style for Software Architecture," unpublished manuscript, May 1994.

**[SG88]** Smith, Gerhart. "STATEMATE and Cruise Control: A Case Study," *Proceedings of COMPSAC 88*, 1988.

**[STARS93]** Software Technology for Adaptable, Reliable Systems (STARS). "Conceptual Framework for Reuse Processes (CFRP), Volume I: Definition, Version 3.0," STARS-VC-A018/001/00, Unisys, October 1993.

**[STARS95]** Software Technology for Adaptable, Reliable Systems (STARS). "Scenarios for Analyzing Architecture Description Languages, Version 1.0," STARS-AC-M001/001/01, Unisys, April 1995.

**[Weid86]** Weiderman, Habermann, Borger, Klein. "A Methodology for Evaluating Environments," *Proceedings of the Second ACM SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments*, Palo Alto CA, December, 1986.